π-calculus Chapter4
Concurrent Processes and Reaction

紙名 哲生
kamina@graco.c.u-tokyo.ac.jp

2000年11月22日

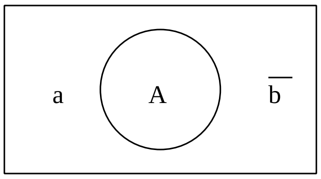
概 要

前回では LTS について strong simulation と strong bisimulation を定義し、sequential process について process expression と structural congruence を定義した。ここでは、並列プロセスについて同様の議論をする。

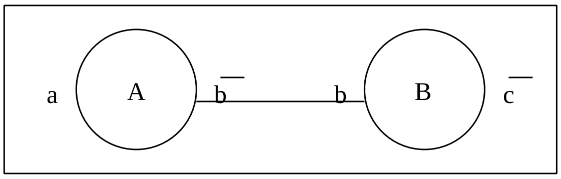
1 Labels and flowgraphs

前回、ラベルを $\mathcal{L} \overset{\text{def}}{=} \mathcal{L} \cup \bar{\mathcal{L}}$ と定義した。ラベルは LTS でのラベルとして用いられていたが、ブラックボックスのボタンとしても使える。a の co-name は $\bar{\bar{a}} \overset{\text{def}}{=} a$ として拡張できる。後でみるように、Act は $\mathcal{L}$ より少しだけ大きい。

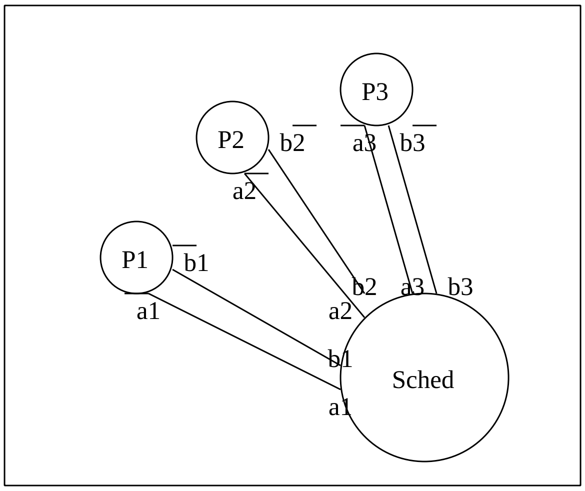
以後、ボタン a、$\bar{b}$ のついたブラックボックス A を以下のように書く。



ブラックボックス同士の interaction を表すために、$(a, \bar{a})$ を用いる。ブラックボックスの動作 a はもう一つのブラックボックスのボタン $\bar{a}$ を押すと考える。これを図で書くと以下のようになる。



これは *flowgraph* の単純な例である。*flowgraph* は遷移ではなく、システムのコンポーネント同士のリンクを表す。前回の sheduler を *flowgraph* で表すと以下のようになる。

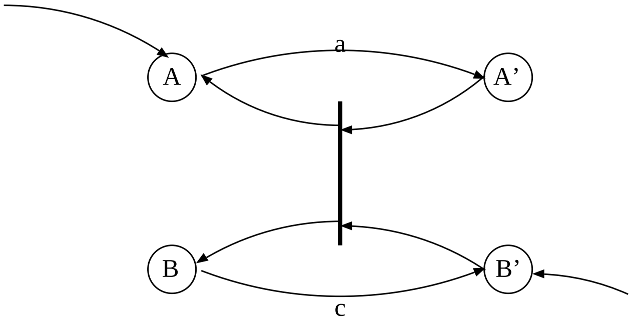


2　Observations and reactions

セクション 1 で出てきたブラックボックス A と B を別々の sequential process として表現すると、

$$
\begin{aligned}
A &\overset{\text{def}}{=} a.A' \quad & B &\overset{\text{def}}{=} b.B' \\
A' &\overset{\text{def}}{=} \bar{b}.A \quad & B' &\overset{\text{def}}{=} \bar{c}.B
\end{aligned}
$$

となる。これを informal に図で表すと以下のようになる。



このような共有された遷移は、1960 年代始めの頃に Carl-Adam Petri によって始めて考え出され、その遷移グラフはペトリネットと呼ばれて有名に

なった。しかしここでは、プロセスが後に述べる composition や restriction によって他のプロセスから構成される様子を図で表現し、プロセス同士のパラメトリックな依存関係を表現する代数的な手法を作りたい。

ここで、b、$\bar{b}$ は**observable** な動作、あるいは**observation** を表現していると考えることができる。b を、それの complement である $\bar{b}$ と interact することによって observe することができる。しかし、遷移そのものは観察することはできないので、これは内部的な動作と考え、**reaction** と呼ぶことにする。reaction は τ と書く。観察できないので、complement はない。

よって動作の集合は $Act \overset{\mathrm{def}}{=} \mathcal{L} \cup \{\tau\}$ となる。今後も Act の要素を $\alpha, \beta, ...$、$\mathcal{L}$ の要素を $\lambda, \mu, ...$ と書くことにする。

3 Concurrent process expressions

Definition 4.1 Concurrent process expression

process expression の集合 $\mathcal{P}$ は以下のように定義される。

$$P ::= A < a_1, ..., a_n > \ \mid \ \sum_{i \in I} \alpha_i.P_i \ \mid \ P_1|P_2 \ \mid \ \mathbf{new} a P$$

和の中の項に M、N を今後よく使う。項の順番は重要ではない。α や $\mathbf{new}a$ の方が和や composition(|) よりも強く結合する。また、$a.b.\mathbf{0}$ と書く代わりに a, b、$\mathbf{new}a\mathbf{new}bP$ と書く代わりに $\mathbf{new}abP$ あるいは $\mathbf{new}a, bP$ と書くことを許すことにする。

restriction $\mathbf{new}aP$ の中で、a を**bound** された名前だという。$\mathrm{fn}(P)$ は bound されてない名前の集合である。bound された名前を別の名前に置き換えることを**alpha-conversion** といい、元のプロセス表現と alpha-conversion によって得られたプロセス式は structually congruent として扱う。

reaction を正式に定義する前に、プロセス P が新しい状態 P' に移るような reaction の例を見てみる。このような reaction を $P \rightarrow P'$ と書くことにする。

Example4.2 Illustrating reaction

セクション 2 の頭の方で登場したプロセス表現を $P = A'|B$ とすると、$P \equiv \bar{b}.A|b.B'$ である。よって b と b' の間に reaction が起こり得る。よって、

$$P \rightarrow A|B'$$

Illustrating alternative reactions

$P = \mathbf{new}a\big((a.Q_1 + b.Q_2)|\bar{a}.\mathbf{0}\big)|(\bar{b}R_1 + \bar{a}.R_2)$ とする。すると、$a.Q_1$ と $\bar{a}.\mathbf{0}$ の間あるいは $b.Q_2$ と $\bar{b}.R_1$ の間のどちらかに reaction が起こり得る。どちらの場合も、和のもう一方の項は捨てられる。以下に述べる Definition4.4($Q_1|\mathbf{0} \equiv Q_1$) を使うと、

$$\begin{array}{rcl} P & \rightarrow & \mathbf{new}aQ_1(\bar{b}.R_1 + \bar{a}.R_2) \\ P & \rightarrow & \mathbf{new}a(Q_2|\bar{a})|R_1 \end{array}$$

の両方が得られる。この例は、reaction によって非決定性が起こることを表している。

$\bar{a}.R_2$ は **new** のスコープの外にあるので、$a.Q_1$ と $\bar{a}.R_2$ の間に reaction は起こらない。これは bound された名前を別の名前に変える (alpha-conversion) 場合において自然なことである。局所的な名前 a を a' に変えると、P は

$$P \equiv \mathbf{new}a'\big((a'.Q'_1 + b.Q'_2)|\bar{a'}|(\bar{b}.R_1 + \bar{a}.R_2)\big)$$

となる。(Q_1 と Q_2 は a を含んでいるかもしれないので、$Q'_1 = \{a'/a\}Q_1$、$Q'_2 = \{a'/a\}Q_2$ とした。)

4 Structural congruence

Definition 4.4 Process context

プロセスコンテクスト $\mathcal{C}$ は、[ ] で表現された hole を含むプロセス表現である。正確には、以下のような文法で定義される:

$$\mathcal{C} ::= [\ ] \mid \alpha.\mathcal{C} + M \mid \mathbf{new}a\mathcal{C} \mid \mathcal{C}|P \mid P|\mathcal{C}$$

$\mathcal{C}[Q]$ はコンテクスト $\mathcal{C}$ の hole をプロセス Q で埋めた結果を表す。基本的なコンテクストは、$\alpha.[\ ] + M$, $\mathbf{new}a[\ ]$, $[\ ]|P$ そして $P|[\ ]$ である。

$\mathcal{C} = [\ ]$ はアイデンティティコンテクストである。$\mathcal{C}[Q] = Q$。

Definition 4.5 Process congruence

$\cong$ を $\mathcal{P}$ 上の同値関係とする。全ての基本的なコンテクストによって保存されるとき、$\cong$ は process congruence であるという。つまり、$P \cong Q$ ならば

$$\begin{array}{rcl} \alpha.P + M & \cong & \alpha.Q + M \\ \mathbf{new}aP & \cong & \mathbf{new}aQ \\ P|R & \cong & Q|R \\ R|P & \cong & R|Q \end{array}$$

Proposition 4.6

同値関係 $\cong$ は全てのコンテクスト $\mathcal{C}$ について、$P \cong Q$ ならば $\mathcal{C}[P] \cong \mathcal{C}[Q]$ のとき、またそのときに限って process congruence である。

$\cong$ は、$\mathcal{C}$ がコンテクスト、$P \cong P' \in \mathcal{E}$ または $P' \cong P \in \mathcal{E}$ のとき、$Q_1, Q_2, ..., Q_n (n \geq 1) s.t. Q_1 = \mathcal{C}[P], Q_{i+1} = \mathcal{C}[P']$ ならば $Q_1 \cong Q_n$ が満

たされるような等式の集合 $\mathcal{E}$ を与えることによっても定義される。つまり、$\mathcal{E}$ の等式を繰り返し適用することによって Q が R に変換できるとき、$Q \cong R$ が成り立つ。

Definition 4.7 Structural congruence

Structural congruence $\equiv$ は、以下の操作によって得られる $\mathcal{P}$ 上の process congruence である:

1. alpha-conversion
2. 和の項の順番を変える
3. $P|\mathbf{0} \equiv P, P|Q \equiv Q|P, P|(Q|R) \equiv (P|Q)|R$
4. $\mathbf{new}a(P|Q) \equiv P|\mathbf{new}aQ \ \ if a \notin \mathrm{fn}(P)$
   $\mathbf{new}a\mathbf{0} \equiv \mathbf{0}, \mathbf{new}abP \equiv \mathbf{new}baP$
5. $A < \vec{b} > \equiv \{\vec{b}/\vec{a}\}P_A \ \ if A(\vec{a}) \overset{\mathrm{def}}{=} P_A$

4. の最初の部分は a を含んでいない部分プロセスは $\mathbf{new}a$ のスコープの中にあっても外にあってもかまわない、ということを意味している。

Definition 4.8 Standard form

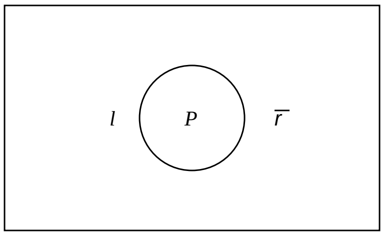
process expression $\mathbf{new}\vec{a}(M_1|...|M_n)$ (各 M_i は空じゃない和) は standard form であるという。($n = 0$ のとき $M_1|...M_n$ は $\mathbf{0}$ を意味する。$\vec{a}$ が空のときは restriction がない場合である。)

Theorem 4.9 全てのプロセスは standard form と structually congruent である。

証明 和の中に入ってない全ての restriction$\mathbf{new}a$ は、$P|\mathbf{new}aQ \equiv \mathbf{new}a(P|Q)$ を使って、4.7 の 3. を併用し、必要ならば alpha-conversion を用いることによって一番外に持っていくことができる。

Example 4.1 Linking

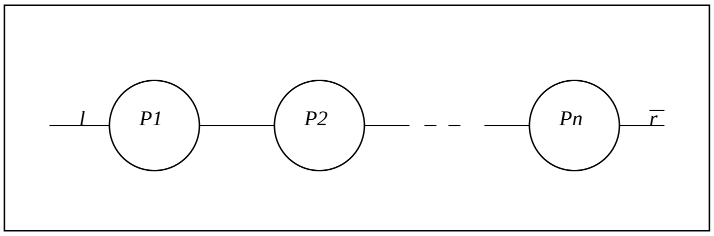
l, r とラベル付けされた二つのポートを持つプロセス P を考える。



このようなプロセスの鎖を考えたい。

linking の二項演算子を以下のように定義できる:

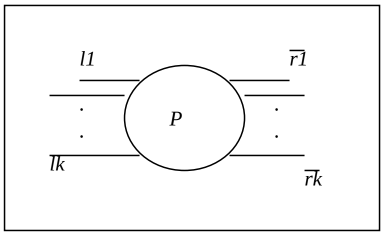
$$P \frown Q \overset{\mathrm{def}}{=} \mathrm{new}m(\{m/r\}P|\{m/l\}Q)$$



m は free ではないので、alpha-conversion を使うことにより、P、Q の中で free でなければどのような名前も使うことができる。

Linking の一般的な定義を与えると次のようになる:

$$P \frown Q \overset{\text{def}}{=} \text{new}\vec{m}(\{\vec{m}/\vec{r}\}P|\{\vec{m}/\vec{l}\}Q)$$



ここで $\vec{m}$ の各要素は distinct で P と Q の中で free でない。linking の定義は $\vec{l}$、$\vec{r}$ に依存している。これらを決定するのにコンテクストを用いることができる。Linking の例として、$V = \{v_1, ..., V_n\}$ という有限データドメインがあったとき、$\vec{l}$ を $v \in V$ が転送されるときのポートとみなすことができる。

Exercise 4.12

$P \frown (Q \frown R) \equiv (P \frown Q) \frown R$

5 Reaction rules

Structural congruence を用いることによって、任意のプロセス間に可能な reaction を二つのルールによって決定することができる。一つは REACT ルールで、Definition 4.13 で説明する。もう一つは TAU ルールで、セクション 2 で登場したものである。どちらの場合も、M や N で表現された和は捨てられる。

これらのルールはプロセスのどこでも起こるというわけではない。例えば、$Q = \bar{a}.(b.B|\bar{b}.C)$ では、$\bar{a}$ が観察されない限り $b.B$ と $\bar{b}.B$ の間の相互作用は起こらない。つまり、

$$a.A|Q \to A|(b.B|\bar{b}.C) \to A|(B|C)$$

である。

PAR と RES は reaction が composition や restriction の中で起きてもよいことを示している。STRUCT は、inference の中で structural congruence を使えることを示している。

Definition 4.13 Reaction

Reaction Rules:

$$\begin{gathered}
\text{TAU :}\tau.P + M \to P \\
\text{REACT :}(a.P + M)|(\bar{a}.Q + N) \to P|Q \\
\text{PAR :}\frac{P \to P'}{P|Q \to P'|Q} \quad \text{RES :}\frac{P \to P'}{\text{new}aP \to \text{new}aP'} \\
\text{STRUCT :}\frac{P \to P'}{Q \to Q'}\ if\ \ P \equiv Q\ and\ P' \cong Q'
\end{gathered}$$

例として、Example 4.3 の例を見てみる。

$$P = \text{new}a\big((a.Q_1 + b.Q_2)|\bar{a}.0\big)|(\bar{b}.R_1 + \bar{a}.R_2)$$

これは、

$$\cfrac{\cfrac{\cfrac{\cfrac{}{(a.Q_1 + b.Q_2)|\bar{a} \to Q_1 0}\text{REACT}}{(a.Q_1 + b.Q_2)|\bar{a} \to Q_1}\text{STRUCT}}{\text{new}a((a.Q_1 + b.Q_2)|\bar{a}) \to \text{new}a.Q_1}\text{RES}}{\text{new}a((a.Q_1 + b.Q_2)|\bar{a})|(\bar{b}.R_1 + \bar{a}.R_2) \to \text{new}a.Q_1|(\bar{b}.R_1 + \bar{a}.R_2)}PAR$$

より $\text{new}a.Q_1|(\bar{b}.R_1 + \bar{a}.R_2)$ が導ける。

Exercise 4.14

$$P \equiv \text{new}a'\Big(\big((a'.Q_1' + b.Q_2')|(\bar{b}.R_1 + \bar{a}.R_2)\big)|\bar{a'}\Big);$$

を示したあと、

$$P \to \text{new}a'((Q_2'|R_1)|\bar{a'})$$

を示す。

もちろん、システム P が先を実行するために、observation が必要な場合もある。このとき、P は**stable** であるという。

Example 4.15 Lottery

ボールの集合 $\{b_1, ..., b_n\}$ からランダムにボールを選び、ボールを取り出したあと同じ動作を繰り返す lottery L を作りたい。内部的な選択を表現するのに τ を用いて、lottery の振舞を

$$Lotspec \overset{\text{def}}{=} \tau.b_1.Lotspec + ... + \tau.b_n.Lotspec$$

のように記述できる。しかし、これではボールの集合のサイズ n 毎に定義が必要である。特定のコンポーネントから、任意の n の lottery を作るために、まず $n = 3$ で試してみる。

最初に、generic cell $A(a, b, c)$ は以下のように定義される:

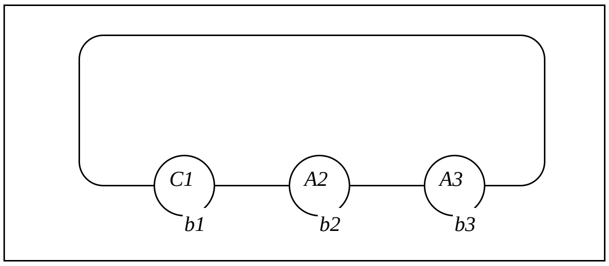
$$A \overset{\text{def}}{=} \bar{a}.C, \;\; C \overset{\text{def}}{=} \tau.B + c.A, \;\; B \overset{\text{def}}{=} b.C$$

動作 $\bar{a}$ は左手のボタンが押されたときに起こる。動作 c は右隣の左手のボタンを押す動作である。次に、

$$\begin{array}{rcl} A_i & \overset{\text{def}}{=} & A < a_i, b_i, a_{i+1} > \\ B_i & \overset{\text{def}}{=} & B < a_i, b_i, a_{i+1} > \\ C_i & \overset{\text{def}}{=} & C < a_i, b_i, a_{i+1} > \end{array}$$

より lottery の三つの状態は以下のように定義される:

$$L_1 = \text{new} a_1 a_2 a_3 (C_1 | A_2 | A_3)$$
$$L_2 = \text{new} a_1 a_2 a_3 (A_1 | C_2 | A_3)$$
$$L_3 = \text{new} a_1 a_2 a_3 (A_1 | A_2 | C_3)$$



Exercise 4.16

Reaction rule を用いて

$$\begin{array}{rcl} L_1 & \rightarrow & L_2 \\ L_1 & \rightarrow & \mathrm{new}\bar{a}(B_1|A_2|A_3) \end{array}$$

を示し、二番目の結果は stable であることを示す。

$\mathrm{new}\bar{a}(B_1|A_2|A_3)$ の状態のとき、外部的な動作 b_1 が起こらなければ次に進めない。

ここで、少し bisimulation $\sim$ にたち戻る。bisimulation では、プロセスが同じ「実行のパターン」を持っていたら二つを同じとみなすものであった。これは、$P \sim Q$ ならば P と Q は interchangeable であることを保証するためであった。ここで、$L < b_1, b_2, b_3 > \sim L < b'_1, b'_2, b'_3 >$ であるが、$b_1 \neq b'_1$ の場合 $L < b_1, b_2, b_3 > |\bar{b1} \not\sim L < b'_1, b'_2, b'_3 > |\bar{b1}$ である。次に、コンカレントプロセスの内部的動作と外部的動作の両方について、LTS を定義する。